Towards an Optimal Implementation of Cascade

Jesus Martinez-Mateo, Christoph Pacher, Alex Ciurana and Vicente Martin

I. Introduction

Cascade [1] is probably the best known protocol for error reconciliation in QKD. Although this is a highly interactive protocol, since it requires many communication between the parties, it is reasonably efficient and easy to implement. Accordingly, a number of modifications and optimizations have been proposed in the literature [2]–[5]. Most of these works concentrate on how to optimize the efficiency of reconciliation by modifying the block length, but others propose modifications to the protocol itself, combining a version of the original Cascade with a second algorithm with a specific focus on improving the reconciliation efficiency [2], or the number of channel communications [3].

The aim of this contribution is to study the modifications of Cascade, comparing them with the original protocol on the grounds of a full set of parameters, so that the effect of these modifications can be fairly assessed. A number of simulations were performed to study not only the efficiency but also other characteristics of the protocol that are important for its practical application, such as the number of communications and the failure probability. Note that, although it is generally believed that the only price to pay for an improved efficiency is an increased interactivity, when looking at all the significant magnitudes a different view emerges, showing that, for instance, the failure probability eliminate some the supposed advantages of these improvements.

Simulation results were initially computed for the original Cascade [1] and the modified version of this protocol proposed in [2]. Correlated pairs of random bit sequences were generated using a congruential pseudo random number generator with common (previously shared) seed. Given the channel parameter Q (i.e., the quantum bit error rate or QBER), errors were generated in one of the sequences simulating independent Bernoulli processes with success probability Q.

II. Cascade protocol

Initially, in [6] the authors propose a protocol for reconciling errors based on a block parity exchange. Two correlated sequences of bit values belonging to different parties are processed in parallel. At each of the parties the sequence is divided into blocks of equal length, then the parity is computed for each block and their values are exchanged through a public noiseless channel. This procedure detects all blocks with parity mismatches. For all those blocks the parties perform a binary search to find and correct one of the errors that have occurred in the block. This procedure detects all odd numbers of errors per block, but corrects only exactly one of them. Thus, the proposed protocol needs to works iteratively. In each successive pass the sequence is shuffled and further parities are exchanged to detect and correct further errors. This protocol is known as BBBSS or BINARY.

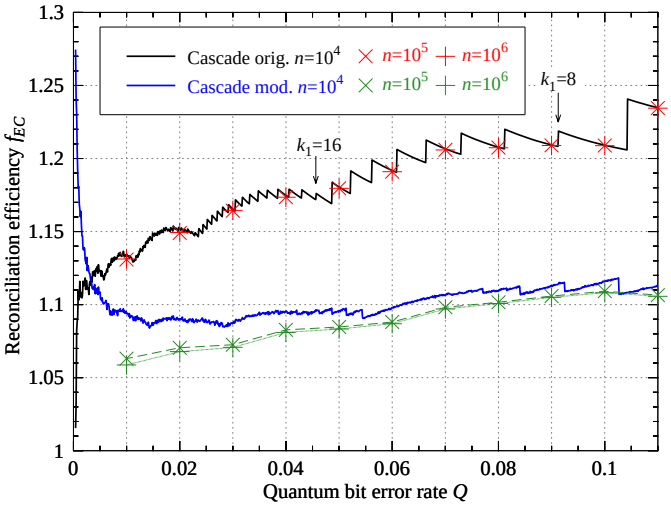

Fig. 1. Average reconciliation efficiency.

Later in [1] they realize that each detected error produces side information that can be used to correct undetected errors of previous passes. The protocol runs for a fixed number of passes. In each pass, the parties divide their sequence into blocks of equal length. The first block size is calculated as a function of the estimated error probability in the quantum channel Q or QBER, $k_1 \approx 0.73/Q$, and it is doubled for successive passes, $k_i = 2k_{i-1}$. Whenever an error is found after the first pass, it uncovers an odd number of additional errors masked in the preceding passes and the algorithm steps back to correct also one of them. Sometimes this correction uncovers another error, starting a cascade of corrections. Therefore this new protocol has been named Cascade. Note that, in a practical implementation of Cascade, blocks and parities are processed in parallel. Therefore, instead of exchange messages with single parities we process and communicate ensembles of parities (i.e., syndromes). Further, note that binary searches (i.e., subblocks parities) are also processed in parallel.

Fig. 1 shows the average reconciliation efficiency as a function of QBER. Simulated results were computed for sequences of 10^4 bits length. We consider this block length as a significant value given that hardware implementations are feasible for this. Results for longer sequences were also simulated but for a limited number of error rates. As shown, Cascade's efficiency does not improve for longer sequences (i.e., short sequences can be corrected as efficiently as longer ones), and the curve exhibit a saw behavior due to changes in

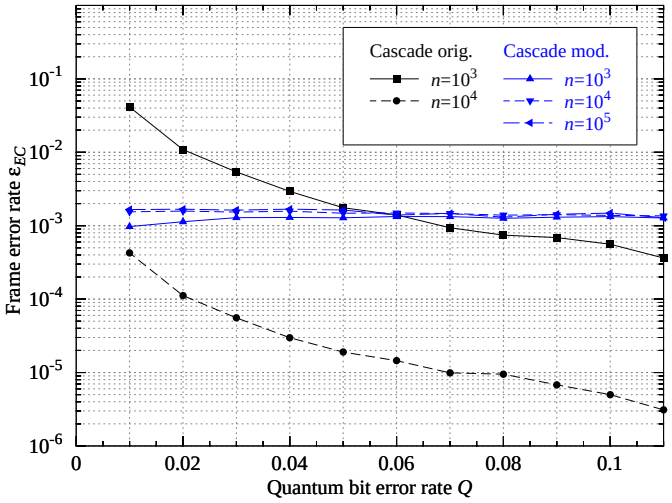

Fig. 2. Failure probability or frame error rate.

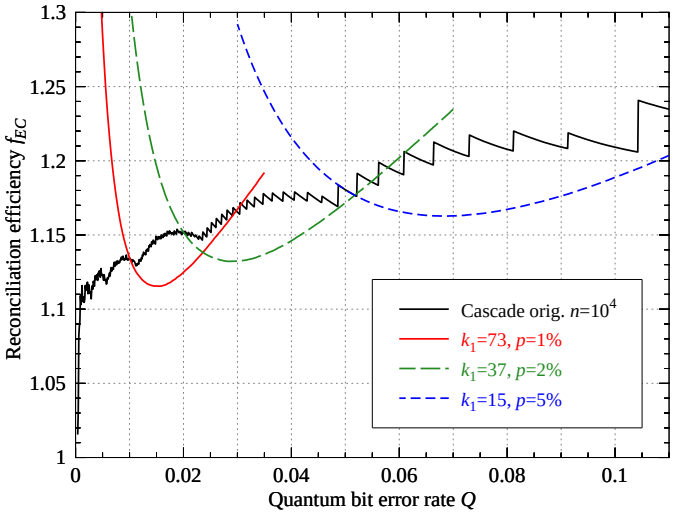

Fig. 3. Average reconciliation efficiency.

the value of the first block size k_1.

Fig. 2 shows the failure probability or frame error rate as a function of QBER. By failure probability we mean the probability that sequences cannot be reconciled given that the sequences belonging to both parties differ by at least one bit.

A. Protocol modification

In [2] the authors propose to perform the first two passes of the original Cascade and later continue with a different algorithm referred as BICONF. The BICONF(s) algorithm works as follows. First, the parties agree on a random subset of bits from their sequences. Then, they compute and exchange the parity value of this subset, and perform two binary searches if the parity differs: one for the chosen subset and other for the complementary subset. This algorithm also works iteratively updating the random subset of chosen bits in each iteration, and it concludes when according to [1] it has been performed s iterations, or according to [2] it has been performed s successive iterations without finding new errors. We have simulated the version described in [2], where the authors propose to run BICONF(s) with $s = 10$. Further, the authors propose to use the following block sizes $k_1 \approx 0.92/Q$ and $k_2 \approx 3k_1$ for the first and second pass of Cascade, respectively. The suggested values should minimize the number of exchanged parities, thus optimizing the reconciliation efficiency.

Fig. 1 shows that the modified protocol improves the efficiency of Cascade, and —in contrast to the original protocol— it further improves, although marginally, for longer sequences. However, this modification has two problems. On the one hand, it does not take into account that a pass of Cascade with block size half of the sequence length (i.e., $k_i = n/2$) operates as BICONF but it can take advantage of the cascade to correct possibly further errors in previous passes. On the other, as shown in Fig. 2, the failure probability is significantly higher. Therefore, the efficiency improve comes at the cost of a higher frame error rate (a fact that is also typical for one-way reconciliation with block codes). The figure also shows that, while the error rate decreases with the block length in Cascade, this is not the case with the modified version where for lengths of 10^5 bits the failure probability remains at 10^{-3}.

B. Parameters optimization

In [4] different values for the first block size and subsequent increases for this were considered and analyzed. According to [4] the efficiency of Cascade is empirically optimal for $k_1 = 0.8/Q$ and $k_i = 5k_{i-1}$. Unfortunately, as in [2], only hundred frames have been simulated which is not enough to empirically verify the failure probability of the proposed protocol, and fairfully compare this with the original Cascade.

We decide then to follow a different direction analyzing the rateless behavior of Cascade—i.e., the ability of the protocol to be adapted to variations in the communication channel. Results were then computed using two different input parameters instead of only QBER: (i) p the error rate value used to calculate the first block size in Cascade, and (ii) Q the actual quantum bit error rate, i.e., the value used to generate discrepancies in the correlated sequences. Note that p may stand for a poor estimate of Q. Therefore, the following simulations show how the protocol behaves under time-varying channel conditions. In addition, as discussed below, these simulations give us information about some parameters used in the protocol (e.g., block sizes) while suggesting possible optimizations.

Fig. 3 shows the average reconciliation efficiency as a function of QBER. Three different cases were considered, for $p = 1\%$, 2% and 5% (i.e., fixing k_1) and variable QBER. The efficiency of Cascade is also depicted in the figure. As expected, it coincides with the new simulations in all the cases when $Q = p$, but surprisingly, the efficiency improves in a range of QBER values when $Q > p$. Again, as expected, a price to pay for a better reconciliation efficiency is a sharp increase in the number of exchanged messages. Also, contrary to what we might expect from the above result, the performance is not significantly affected regarding the failure

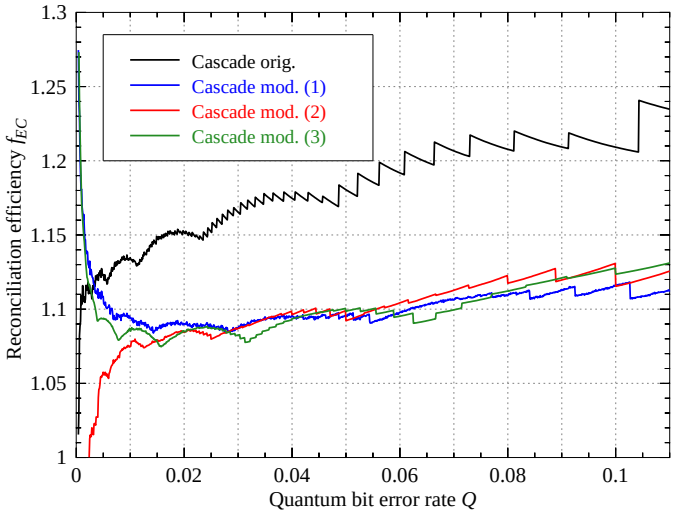

Fig. 4. Average reconciliation efficiency.

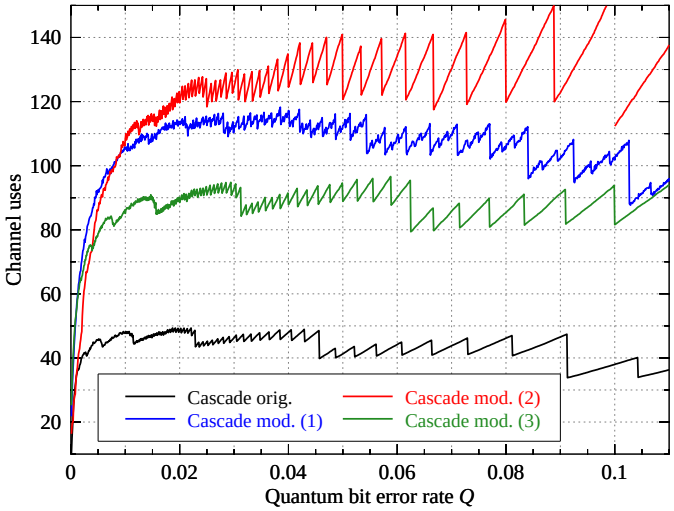

Fig. 5. Communication rounds.

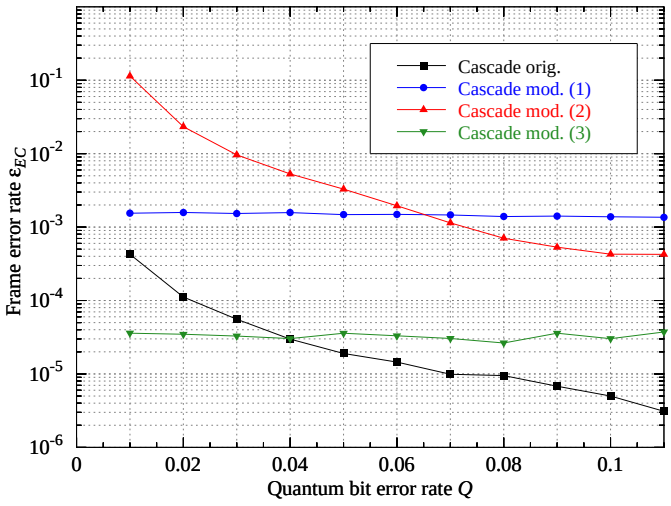

Fig. 6. Failure probability or frame error rate.

probability (see accompanying long write-up). In consequence, these results clearly show that the original Cascade protocol may be improved just by updating the first block size. The efficiency of Cascade is thus presumably optimal for $k_1 \approx 1/Q$. In other words, the results suggest using as the first block size the one that contains one error, on average. Note that this was originally suggested in [5]. Its purpose is to maximize the number of errors corrected during the first pass.

III. Proposed Optimization

Based on the above results the behavior of Cascade with a optimal first block size $k_1 = 1/Q$ and $k_2 = 2k_1$ is studied. Note that we might mistakenly infer that the optimal value for the second block size should be calculated similarly to the first one. However, when we run the protocol with this criterion we quickly realize that the efficiency worsens. We conclude then that, while larger block sizes are less able to correct errors, these allow to better follow the cascade and correct further errors using the first block size. Unfortunately, with the current theoretical framework we have no way to optimize this second block size. This optimization is proposed for sequences of 10^4 bits length, adequate for hardware implementations. Thus, taking into account that the simulated results suggest sizes for the next blocks exceeding the half of the sequence length[1], the value used for k_i is fixed to $k_i = \lceil n/2 \rceil$ for $i > 2$.

Fig. 4 shows the average reconciliation efficiency as a function of QBER. Cascade is compared to three modified versions: (1) the modified protocol proposed in [2], (2) the version with the optimized parameters suggested in [4], and (3) the version proposed here with 16 passes. As shown, the efficiency is similar for the three proposed optimizations, corresponding to approximately half of the efficiency of Cascade. As shown in Fig. 5, all the optimizations exceed the number of communication rounds of the original protocol, but the proposed here is the one with fewer communications despite having undergone 16 passes. Finally, the failure probability is shown in Fig. 6. Again, these results show that the optimization proposed here also improves the others, with a constant failure probability (i.e., independent of the QBER), and similar or better than the original Cascade in a broad QBER range.

[1]Once concluded the second pass, the residual error ε'_{EC} suggests using block sizes greater than the half of the frame length, i.e., $1/\varepsilon'_{EC} > n/2$.

References

[1] G. Brassard and L. Salvail, "Secret-key reconciliation by public discussion," in *Eurocrypt'93*, vol. 765, 1994, pp. 410–423.

[2] T. Sugimoto and K. Yamazaki, "A study on secret key reconciliation protocol "cascade"," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E83-A, no. 10, pp. 1987–1991, 2000.

[3] W. T. Buttler *et al.*, "Fast, efficient error reconciliation for quantum cryptography," *Physical Review A*, vol. 67, no. 5, p. 052303, May 2003.

[4] H. Yan *et al.*, "Information reconciliation protocol in quantum key distribution system," in *ICNC 2008*, vol. 3, 2008, pp. 637–641.

[5] R. N. Ii-Yung, "A probabilistic analysis of binary and cascade," unpublished manuscript.

[6] C. H. Bennett *et al.*, "Experimental quantum cryptography," *J. Cryptology*, vol. 5, no. 1, pp. 3–28, 1992.